Related to other papers in this special issue	29 (p285); 8 (p78)
Addressing FAIR principles	F1, F2, F3, F4, A1, A1.1, A1.2, A2, I1, I2, I3, R1, R1.1, R1.2, R1.3

FAIR Computational Workflows

Carole Goble[1†], Sarah Cohen-Boulakia[2], Stian Soiland-Reyes[1,4], Daniel Garijo[3], Yolanda Gil[3], Michael R. Crusoe[4], Kristian Peters[5] & Daniel Schober[5]

[1]Department of Computer Science, The University of Manchester, Oxford Road, Manchester M13 9PL, UK

[2]Laboratoire de Recherche en Informatique, CNRS, Université Paris-Saclay, Batiment 650, Université Paris-Sud, 91405 ORSAY Cedex, France

[3]Information Sciences Institute, University of Southern California, Marina Del Rey CA 90292, USA

[4]Common Workflow Language project, Software Freedom Conservancy, Inc. 137 Montague St STE 380, NY 11201-3548, USA

[5]Leibniz Institute of Plant Biochemistry (IPB Halle), Department of Biochemistry of Plant Interactions, Weinberg 3, 06120 Halle (Saale), Germany

Keywords: Computational workflow; Reproducibility; Software; FAIR data; Provenance

Citation: C. Goble, S. Cohen-Boulakia, S. Soiland-Reyes, D. Garijo, Y. Gil, M.R. Crusoe, K. Peters & D. Schober. FAIR computational workflows. Data Intelligence 2(2020), 108–121. doi: 10.1162/dint_a_00033

ABSTRACT

Computational workflows describe the complex multi-step methods that are used for data collection, data preparation, analytics, predictive modelling, and simulation that lead to new data products. They can inherently contribute to the FAIR data principles: by processing data according to established metadata; by creating metadata themselves during the processing of data; and by tracking and recording data provenance. These properties aid data quality assessment and contribute to secondary data usage. Moreover, workflows are digital objects in their own right. This paper argues that FAIR principles for workflows need to address their specific nature in terms of their composition of executable software steps, their provenance, and their development.

† Corresponding author: Carole Goble (E-mail: carole.goble@manchester.ac.uk, ORCID: 0000-0003-1219-2137).

1. INTRODUCTION

In data intensive science, e-infrastructures and software tool-chains are heavily used to help scientists manage, analyze, and share increasing volumes of complex data [1]. Data processing tasks like data cleansing, normalisation and knowledge extraction need to be automated stepwise in order to foster performance, standardisation and re-usability. Increasingly complex data computations and parameter-driven simulations need reliable e-infrastructures and consistent reporting to enable systematic comparisons of alternative setups [2, 3]. As a response to these needs, the practice of performing computational processes using workflows has taken hold in different domains such as the life sciences [4, 5, 6], biodiversity [7], astronomy [8], geosciences [9], and social sciences [10]. Workflows also support the adoption of novel computational approaches, notably machine learning methods [11], due to the ease with which single components in a processing pipeline can be exchanged or updated.

Generally speaking, a workflow is a precise description of a procedure – a multi-step process to coordinate multiple tasks and their data dependencies. In computational workflows each task represents the execution of a computational process, such as: running a code, the invocation of a service, the calling of a command line tool, access to a database, submission of a job to a compute cloud, or the execution of data processing script or workflow. Figure 1 gives an example of a real workflow for variant detection in genomics, represented using the Common Workflow Language[①] open standard [12].

Computational workflows promise support for automation that scale across computational infrastructures and large datasets while shielding users from underlying execution complexities such as inter-resource incompatibilities and software dependencies. From an execution perspective, workflows are a means to handle the work of accessing an ecosystem of software and platforms, managing data, securing access, and handling heterogeneities. From a reuse and reproducibility perspective, the automated methods can be packaged and ported across computational platforms, easing how we can create and execute workflows in different environments and across the diverse expertise levels of users. From a reporting perspective, they are a means to specify and document the experiment design and report the methodology: accurately recording the data inputs, parameter configurations and history of their runs and the provenance of their output data products [13]. The provenance of a result (i.e., why and how a given result has been obtained by an analysis) enables the comprehension, comparison and verification of multiple results, and hence facilitates the exchange, standardisation, trust and reusability of those results.

The rise in the use of workflows has been accompanied by a range of diverse systems by which they can be implemented. At one end of the spectrum reside ad-hoc scripts (shell code, Python, Java, etc.) and interactive notebooks which provide an intuitive interface to quickly interact with the analysis results (e.g., Jupyter[②], RStudio[③], Zeppelin[④]). At the other end are *Workflow Management Systems* (WfMS) that provide

① http://commonwl.org/.
② https://jupyter.org/.
③ https://www.rstudio.com/.
④ https://zeppelin.apache.org/.

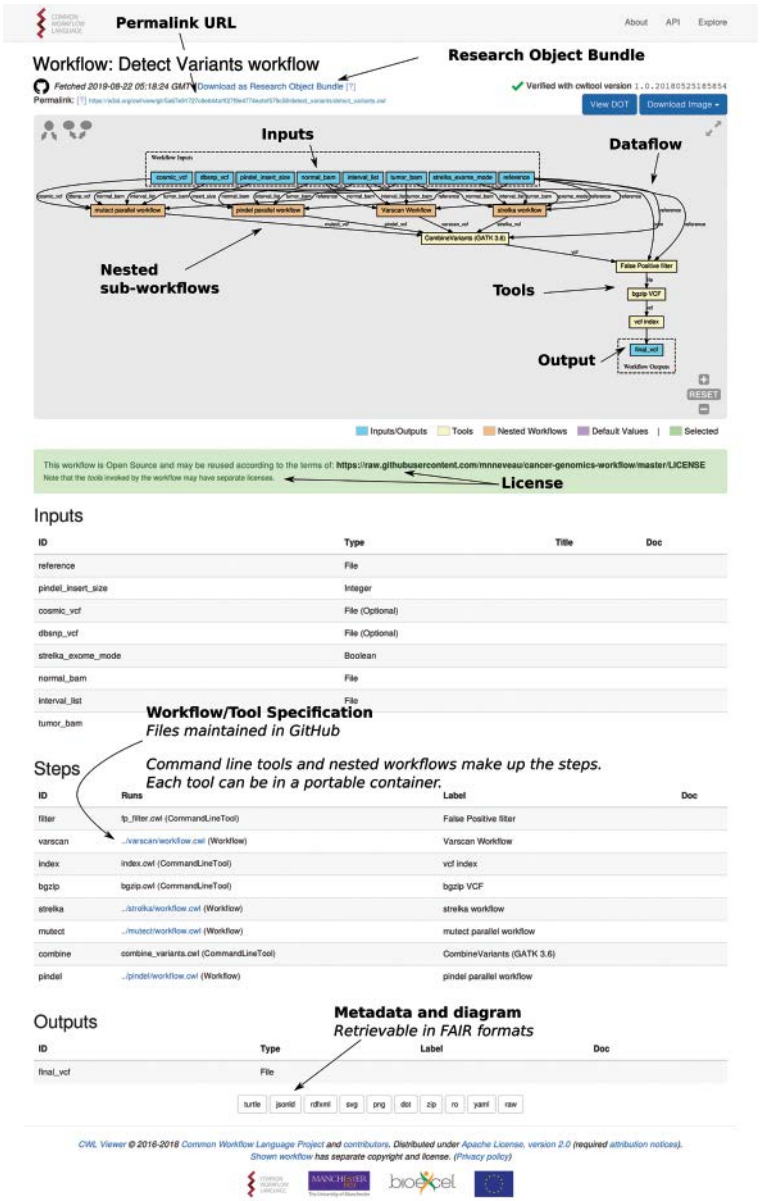

Figure 1. A workflow for detecting variants in genome sequences[5]. The workflow is specified in the Common Workflow Language, displayed using the CWL Viewer. Any CWL compliant WfMS execution engine should be able to execute this standardized workflow description and obtain the same results independent of their underlying infrastructure, for instance using Toil[6] executed on SLURM, or REANA[7] on Kubernetes.

[5] https://w3id.org/cwl/view/git/5a67e91727c8eb44afff27f9e4774eafef579c58/detect_variants/detect_variants.cwl.

[6] http://toil.ucsc-cgl.org/.

[7] https://reana.readthedocs.io/.

a feature-rich infrastructure for the definition, set-up, execution and monitoring of a workflow. Some WfMS are aimed at general applications (e.g. KNIME[8] [11]) whilst others have been adopted by specific communities with specialised features and component collections (e.g. Nipype[9] for neuro-bioimaging [14]). Different aspects of FAIR principles will apply across this range of implementation choices.

WfMS can roughly be divided into two types, namely *coarse-grained*, with a prime focus on chaining locally hosted or distributed tools (e.g. Galaxy[10] [15], KNIME [11], Taverna[11] [16]) and *fine-grained* focusing on optimising computational resources over Distributed Computing Infrastructures (DCI) or High Performance Computing (HPC) for applications (e.g. Pegasus[12] [17], SnakeMake[13] [18], Nextflow[14] [19], Dispel4Py[15] [20]) and cloud-based container orchestration (Kubernetes[16]). Many WfMS mix the two kinds [21]. All WfMS aim to handle common cross-cutting concerns on behalf of the workflow execution. Concerns include: *resource scalability* (optimisation, concurrency and parallelisation), *secure execution* (of tools in their environment, monitoring and fault handling), *tracking* (process logging and data provenance tracking) and *data handling* (secure access, movement, reference management). WfMS vary in how their users interact with them, for example by ABIs/APIs, scripting or command lines, or a GUI using "drag, drop and linking". WfMS may execute over HPC or geographically distributed clusters, cloud environments across systems, or even from desktops. They consequently vary in their mechanisms to prepare their components to become executable steps and must manage portability and dependencies on the infrastructure used to run them.

Computational workflows are composed of modular building blocks that have been prepared with standardised interfaces to be linked together and run by a computational engine. Thus, the key characteristic is *the separation of the workflow specification from its execution*. Capturing the control flow order between components explicitly exposes the dataflow and data dependencies between the inputs and outputs of the processing steps. This explicit separation is fundamental to supporting workflow comprehension, design modularity, workflow comparisons and alternative execution strategies. Scripting environments tend to interleave data and computational processes, although systems such as YesWorkflow [22] provide the means to annotate existing scripts with special comments that reveal their hidden computational modules and dataflows. Interactive notebooks make the distinction when organized appropriately by defining their dataflow in the form of interactive computational cells; i.e., input and output variables are explicit *in* each cell, data dependencies are explicit *on* each cell, and the steps are executed in order. Notebooks can also be used as "meta workflows" when steps run a script or a WfMS.

[8] https://www.knime.com/.
[9] https://nipype.readthedocs.io/.
[10] https://galaxyproject.org/.
[11] https://taverna.incubator.apache.org/.
[12] https://pegasus.isi.edu/.
[13] https://snakemake.readthedocs.io/en/stable/.
[14] https://www.nextflow.io/.
[15] https://pypi.org/project/dispel4py/.
[16] https://kubernetes.io/.

We propose that FAIR Principles apply to workflows, and WfMSs, in two major areas:

- ***FAIR data**:* Properly designed workflows contribute to FAIR data principles, since they provide the metadata and provenance necessary to describe their data products and they describe the involved data in a formalized, completely traceable way.
- ***FAIR criteria for workflows as digital objects**:* Workflows are research products in their own right, encapsulating methodological know-how that is to be found and published, accessed and cited, exchanged and combined with others, and reused as well as adapted.

These two aspects are explored in the rest of the article. References to FAIR principles [23] are given in brackets.

2. FAIR DATA FOR AND FROM WORKFLOWS

Computational workflows are enablers of *automated* data processing. For automation to be most effective the data the workflows act on should be FAIR: unique resolution of identifiers, explicit data organisations, structures and semantics, machine-readable licenses and access permissions, high data quality and so on. FAIR data would enable a WfMS to automatically make *informed choices* from the phase of the workflow design (e.g., by suggesting tools fitting data features when several alternative tools could be considered for a given data analysis step) to the phase of workflow execution (e.g., by validating the data against a step's expected type). A WfMS needs to be able to access precise information on data origin, the way of accessing it, and a set of associated metadata, which is ideally described by established vocabularies and computer-interpretable semantics.

Research domains such as the life sciences have developed open ontologies, vocabularies and services for data interoperability (I1, I2, I3) and identifier resolution (F1, A1). Efforts such as the Breeding API (BrAPI)[©] standardise the interface for exchanging data between applications and the EDAM ontology[®] [24] precisely specify the input and output of tools executed in a workflow (see FAIRsharing.org [25] for examples). Formalised and fine grained annotation of data is still considered costly to produce. Consequently, a significant amount of workflow processing still deals with metadata wrangling, format transformations and identifier mapping [26].

Workflows are in turn key contributors to FAIR data compliance. In a world of expanding and diversifying processing tools and computational operating environments, they encode standardised data practices and capture formal computer-interpretable provenance data. The workflow specification itself can be thought of as a recipe to produce a data product that exposes the extent of the effort made to make the data FAIR and it is formal enough to be validated against emerging FAIR indicators [27]. Determining whether the data produced by a workflow is FAIR is not straightforward and requires concrete criteria, which should be provided by both the FAIR indicators and the workflow specification.

© https://brapi.docs.apiary.io/.
® http://edamontology.org/.

The combination of FAIR data and FAIR tools within a supportive FAIR e-infrastructure would significantly aid the operation and quality assurance of workflows. Machine processable metadata describe the inputs, outputs and performance of tools and the underlying resources for running workflows and managing results. Examples include annotations on tools and libraries (e.g. Bio.tools[®], Bioconductor, CRAN, PyPI) and on software containers (e.g. Biocontainers[②]). Standardised specifications on handling data formats and executables, automated handling of tool dependencies, versioning and other explicit metadata on computational resource-needs would aid harmonisation of successive software tools execution and efficient job scheduling and data movement throughout different FAIR e-infrastructures.

For data generation a standardised workflow specification and automated execution contributes to transparency, reproducibility, analytic validity, quality assurance and the attribution and comparison of results. Well-designed workflow management systems can automate the production of metadata descriptions of data products (F2, I2, I3, R1.3) and the deposition of data in searchable resources (F4). Identifiers, licensing and access present interesting challenges in workflow execution:

- *Identifiers (F1, F3, A1):* concerns include the propagation of identifiers through the workflow, tracking data attribution [28] and the minting of identifiers for large numbers of intermediate results. Minids [29] are proposed as light-weight identifiers to unambiguous name, identify and reference research data products. Identifiers can then optimise data exchange by reference and reduce unnecessary or insecure data movements. Workflows need to move data references through their engines and not the data itself.
- *Licensing (R1.1):* As workflows often access and combine data from different sources, data licenses need to be respected, honoured and propagated, as do licenses on the software used by the workflow tasks. Combining licenses is particularly tricky and can impact on the ability to license the workflow itself or its data products, often neglected or delayed because of this challenge.
- *Data access (A1.1, A1.2):* Single sign-on for workflow constituents requires harmonised Authentication and Authorization Infrastructure (AAI) propagation through the different tasks, which may be hosted by different service providers using different systems.

Workflows intrinsically provide precise documentation of how the data has been generated (R1.2). The details of every executed process together with comprehensive information about the execution environment used to derive a specific data product is *retrospective provenance*, either observed by the WfMS or disclosed to it by the computational task itself. A great deal of work has focused on provenance tracking of computational workflows [30], leading to standardisation efforts such as the W3C PROV model and ontology [31] (I1, I2). Challenges remain: provenance standards have yet to be fully embraced by WfMS, there are still shortages of provenance processing tools. Automated provenance collection can be too fine-grained and too detailed to be of service to researchers [32], indicating a need for ontological abstractions. So, the computational steps are themselves *unFAIR*. Although open source tools allow us to inspect procedures, many codes are proprietary software or opaque boxes (especially those only available as

[®] http://bio.tools/.
[②] https://biocontainers.pro/.

run-time binaries or representing deep learning algorithms) that do not disclose the link between their inputs and outputs, breaking the provenance lineage of data. Steps in coarse-grained workflows are often wrapped applications with buried sub-workflows and manual (not tracked) steps within. Data resources and tools do not always report basic metadata such as their version or licence in a standardised, machine interpretable way. Bioschemas[29] aims to get such metadata marked-up in resources in a lightweight way. A greater problem is unFAIR service provision, whereby the components change their interfaces without notice, breaking the workflows that use them. Given their data focus, the FAIR principles are chiefly focused on the availability of metadata rather than the quality of service of the databases, tools and the e-infrastructures in which data can exist.

3. FAIR CRITERIA FOR WORKFLOWS AS DIGITAL OBJECTS

The initial FAIR criteria have been envisioned for data. As workflows are digital objects in their own right, it is natural to draw an analogy with data and to try to apply the FAIR Principles to them. The majority of workflows are not yet registered in specialised repositories or are stored in software repositories indistinguishable from other software. Conventions for naming workflows still have to be devised (F1). Workflows vary in the quality of their documentation, as happens with software, described using proprietary or native programming languages.

Researchers have been actively exploring ways for workflows to be FAIR. Workflow registries and repositories typically cater for specific WfMSs, such as KNIMEHub[30] for KNIME and nf-core[31] for Nextflow Life Science pipelines, to support findability and accessibility (F4), with description and metadata associated with deposited workflows (F2) and in some cases persistent, unique identifiers (F1). Access is typically baked into the workflow systems (A1), for example only Galaxy workflows are available in dedicated Galaxy installations such as Workflow4Metabolomics[32]. Others such as WINGS[33] provide means to export workflows augmented with semantic representations as Linked Data [33]. Workflow findability in repositories has been studied [34] alongside workflow similarity [35] where workflows are compared based on their metadata and structure. For workflows to be accessible in the same way as data, they need to be archived and cited just as data is archived and cited using citation metadata [36]. In the schema.org mark-up used by citation infrastructures such as Datacite, terms indicate data that is derived from other data. Ideally there should also be terms indicating the software or service used to perform a transformation, harmonised with workflow provenance.

myExperiment[34] [37] is an attempt at a WfMS agnostic repository, pioneering approaches for workflow finding, sharing and publishing with licenses. It credits authors when workflow designs were reused or

[29] http://bioschemas.org/.
[30] https://hub.knime.com/.
[31] https://nf-co.re/.
[32] https://workflow4metabolomics.org/.
[33] http://www.wings-workflows.org/.
[34] http://myexperiment.org/.

repurposed, and packages workflows into collections and with other digital objects such as associated data files and publications. The work laid the foundations for workflow based Research Objects[@] [38] that allows for bundling of all the artefacts associated with an investigation or piece of research into one whole that can also be cited. Figure 1 workflow's description files and links to executable containers and data files can be downloaded in a Research Object zip-based bundle along with citation metadata and assigned a DOI on deposit. The European Open Science Cloud for Life Sciences[@] has started work to build a workflow registry (F4) using the CWL standards with Research Objects federated (I3) with registries for tools (bio.tools) and containers (Biocontainers).

Several attempts have been made to standardise workflow descriptions in order to aid discoverability (F1) and enable interoperability (I1). The Interoperable Workflow Intermediate Representation [39] was proposed as a common bridge for translating fine-grain workflows between different languages, independent of the underlying distributed computing infrastructure [40]. The Workflow Description Language[@] and the Common Workflow Language [12] are recent community efforts to describe workflows. The CWL open standards are used to describe workflows and command-line tool interfaces in a way that makes them portable and scalable across a variety of software and hardware environments and runnable by other CWL-compliant engines. This last point is critical. As *descriptions* of processes workflows inherit properties of FAIR data, but as *executable* processes they inherit properties of *software*. Workflows as processes challenge the FAIR principles by their structure, forms, versioning, executability, and reuse.

Structure. Workflows are often inherently composite. Their components can be workflows in a nested, fractal way, i.e., (interdependent or sets of) sub-workflows that can be executed as part of complex workflows (see Fig.1). The distinction between a workflow and its component steps is blurred [41]. FAIR principles can thus be applied simultaneously on multiple levels. To render composite workflows and sub-workflows findable relies also on the findability of the involved tools and data types as researchers often use these as search attributes. FAIR properties on the components – metadata, licensing, author credit, access authorization and so on – propagate to the workflow level and may be incompatible. Fundamentally, how we identify, cite and credit *composite, multi-authored* objects is an open question [42].

Forms. When we speak of a FAIR workflow what do we mean? A workflow can be a CWL specification with test or exemplar data; an implementation of that design in a WfMS; an instantiation of that implementation ready to be run with input data and parameters set and computational services spun up; a run result with intermediate and final data products and provenance logs [33]. Workflow-centric Research Objects attempt to create a metadata framework to capture and aggregate each form, but each may have different FAIR criteria.

[@] http://researchobject.org/.
[@] http://www.eosc-life.eu/.
[@] https://software.broadinstitute.org/wdl/.

Versioning. Like software, workflows are living artefacts to be maintained, updated, and eventually deprecated. The code components, the WfMS itself and the underlying computational infrastructures they run on evolve and change. The evolution of a workflow is a form of provenance (R1.2) that tracks any alteration of an existing workflow, resulting in another version that may produce the same or different results [43]. Moreover, to make methodological variants, workflows can be recycled and repurposed: cloned, forked, merged and dramatically changed. Workflow repositories such as nf-core[23] embrace this software nature, building on top of collaborative development environments such as github that natively support versioning as well as testing and validation. Thus, FAIR principles have to address versioning and "fixivity"; that is, the need to snapshot a workflow and its dependencies to fix its reproducible state and associate a persistent identifier.

Executability. Workflows are executable objects. To be interoperable and reusable they need to be *portable*, encapsulating all their runtime dependencies. Lightweight container-based virtualisation solutions to distribute software (e.g. Docker[⑱], Singularity[⑲]) and platform independent software packaging and distribution (e.g. Conda[⑳]) revolutionise workflow reusability. Nevertheless, workflows and the software tools they use are time limited objects whose active lifespan is dependent on that of their components and WfMS as much as on their scientific relevance. Consequently CWL addresses both explicit support for containerised execution and the lifting of workflow descriptions from the WfMS or application it is embedded in so that it may be runnable in other CWL-compliant engines, even when no longer executable in its native form. In workflow e-infrastructures (e.g. in local workstations or cloud environments), resource limitations need to be defined by the workflow. The stability of the execution environment needs to be covered by the infrastructure and its components. Security and access control issues are crucial as workflow systems often execute codes in shared distributed resources.

Reuse. Reusing workflows turns out to mean different things depending on the purpose of the reuse. *Workflow redo/re-run* re-executes the exact same workflow, data, parameter settings and tools with the aim to re-create identical results for testing the robustness of the process. *Workflow replication* allows minor changes, usually in the workflow environment and/or parameter settings, but the results are expected to be the same. *Workflow reproduction* aims to retain the same analysis but with variations to the means (steps) or data to test the robustness of the results. *Workflow reuse* may use all or part of the original workflow with new data, with a possible different aim in *workflow repurposing/recycling* [33, 44]. Regardless of intent, the workflow user must be confident that the expected results are generated. R1 is fostered by robust software practices [45, 46, 47] that entail:

- Proper testing of the computational workflow and its modules as well as the software tools that are invoked during workflow runtime;
- Validation of interoperability claims that tests workflow replication on different platforms;

⑱ https://www.docker.com/.
⑲ https://sylabs.io/singularity/.
⑳ https://conda.io/.

- Validation of parameters to preclude workflow failure and faulty or unsafe results. The formulation of parameters must therefore be FAIR and must include documentation and explanation of their purpose and range definitions (testing of parameter ranges). The BioCompute Object specification [48] emphasises detailed representation and validation of parameters for regulatory approval of reusable computational pipelines for precision medicine.

These thoughts lead to two conclusions. First that treating FAIR workflows as data artefacts only goes so far. Their characteristics as software artefacts means that they should also be subject to appropriate FAIR principles for software, incorporating best practices for software maintainability, maturity and computation reproducibility [49]. Second, that the individual parts, forms, versions and execution environments of a workflow need to be FAIR by themselves, leading to complex interdependencies which need to be covered by additional FAIR metrics aligned to their nature. Generally a compromise has to be found between the amount of work users have to invest in annotating their workflow and the amount of metadata needed for verification by FAIR indicators. Automatic annotation strategies and annotation tool support will be crucial to ease the burden of workflow developers.

4. CONCLUSIONS

Computational workflows capture complex multi-step methods that require FAIR practices in order to be properly published, findable, accessed, cited, reused, and combined with others. FAIR principles for data, and for software, are generally applicable, but need to be extended in order to address the processual nature of workflows. Consequently new FAIR indicators will also need to be developed. A framework for FAIR workflows will enhance reproducibility, quality and transparency of the data generated, but also of the processing path that lead to the data results. When used to document the provenance of new data products, workflows become a powerful component for FAIR data practices and provide new capabilities such as better findability, ideally based on the intrinsic methods used to generate data. The FAIRification of workflows along all these lines will pave the way for trustable data with the added value of being ready for secondary data reuse and exploitation by third parties.

AUTHOR CONTRIBUTIONS

C. Goble (carole.goble@manchester.ac.uk, corresponding author) and D. Schober (Daniel.Schober@ipb-halle.de) initiated the effort and conceived the paper. C. Goble co-ordinated and led the writing, and edited the manuscript. C. Goble, S. Cohen-Boulakia (cohen@lri.fr), S. Soiland-Reyes (soiland-reyes@manchester.ac.uk), D. Garijo (dgarijo@isi.edu), Y. Gil (gil@isi.edu), M.R. Crusoe (michael.crusoe@gmail.com), K. Peters (Kristian.Peters@ipb-halle.de), D. Schober all contributed to the concepts, arguments, and written text. All reviewed the text. D. Schober and M.R. Crusoe contributed examples.

ACKNOWLEDGEMENTS

Carole Goble acknowledges funding by BioExcel2 (H2020 823830), IBISBA1.0 (H2020 730976) and EOSCLife (H2020 824087). Daniel Schober's work was financed by Phenomenal (H2020 654241) at the initiation-phase of this effort, current work in kind contribution. Kristian Peters is funded by the German Network for Bioinformatics Infrastructure (de.NBI) and acknowledges BMBF funding under grant number 031L0107. Stian Soiland-Reyes is funded by BioExcel2 (H2020 823830). Daniel Garijo, Yolanda Gil, gratefully acknowledge support from DARPA award W911NF-18-1-0027, NIH award 1R01AG059874-01, and NSF award ICER-1740683.

REFERENCES

[1] M. Atkinson, S. Gesing, J. Montagnat & I. Taylor. Scientific workflows: Past, present and future, Future Generation Computer Systems 75(2017), 216–227. doi: 10.1016/j.future.2017.05.041.

[2] E. Deelman, T. Peterka, I. Altintas, C.D. Carothers, K.K. van Dam, K. Moreland, … & J. Vetter. The future of scientific workflows. The International Journal of High Performance Computing Applications 32(1)(2017), 159–175. doi: 10.1177/1094342017704893.

[3] K. Peters, J. Bradbury, S. Bergmann, M. Capuccini, M. Cascante, P. de Atauri, … & C. Steinbeck. PhenoMeNal: Processing and analysis of metabolomics data in the cloud. GigaScience 8(2)(2019). doi: 10.1093/gigascience/giy149.

[4] S. Cohen-Boulakia, K. Belhajjame, O. Collin, J. Chopard, C. Froidevaux, A. Gaignard, … & C. Blanche. Workflows for computational reproducibility in the life sciences: Status, challenges and opportunities. Future Generation Computer Systems 75(2017), 284–298. doi: 10.1016/j.future.2017.01.012.

[5] D. Garijo, O. Corcho, Y. Gil, M.N. Braskie, D. Hibar, X. Hua, N. Jahanshad, P. Thompson & A.W. Toga. Workflow reuse in practice: A study of neuroimaging pipeline users. In: Proceedings of the IEEE Conference on e-Science, Guarujua, Brazil, 2014. doi: 10.1109/eScience.2014.33.

[6] A. Shade & T.K. Teal. Computing workflows for biologists: A roadmap. PLOS Biology 13(11)(2015). doi: 10.1371/journal.pbio.1002303.

[7] C. Mathew, A. Güntsch, M. Obst, S. Vicario, R. Haines, A. Williams, Y. de Jong & C. Goble. A semi-automated workflow for biodiversity data retrieval, cleaning, and quality control. Biodiversity Data Journal 2(2014), e4221. doi: 10.3897/BDJ.2.e4221.

[8] W. Freudling, M. Romaniello, D.M. Bramich, P. Ballester, V. Forchi, C. E. García-Dabló, S. Moehler & M. J. Neeser, Automated data reduction workflows for astronomy: The ESO Reflex environment, Journal Astronomy and Astrophysics 559(2013). doi: 10.1051/0004-6361/201322494.

[9] C. Duffy, Yo. Gil, E. Deelman, S. Marru, M. Pierce, I. Demir & G. Wiener. Designing a road map for geoscience workflows. Eos93(24)(2012), 225–226. doi: 10.1029/2012EO240002.

[10] K.J. Turner & P.S. Lambert. Workflows for quantitative data analysis in the social sciences. International Journal on Software Tools for Technology Transfer 17(3)(2015), 321–338. doi: 10.1007/s10009-014-0315-4.

[11] M.R. Berthold, N. Cebron, F. Dill, T.R. Gabriel, T. Kötter, T. Meinl, P. Ohl, C. Sieb, K. Thiel & B. Wiswedel. KNIME: The Konstanz Information Miner. In Data Analysis, Machine Learning and Applications. Studies in Classification, Data Analysis, and Knowledge Organization. Berlin: Springer. doi: 10.1051/0004-6361/201322494.

[12] P. Amstutz, M.R. Crusoe, N. Tijanić. (editors) B. Chapman, J. Chilton, M. Heuer, A. Kartashov, Da. Leehr, H. Ménager, M. Nedeljkovich, M. Scales, S. Soiland-Reyes & L. Stojanovic. Common workflow language, v1.0.

Specification. Common Workflow Language working group (2016). Available at: https://w3id.org/cwl/v1.0/. doi: 10.6084/m9.figshare.3115156.v2.

[13] V. Cuevas-Vicenttín, S. Dey, S. Köhler, S. Riddle & B. Ludäscher. Scientific workflows and provenance: Introduction and research opportunities. Datenbank Spektrum 12(3)(2012), 193–203. doi: 10.1007/s13222-012-0100-z.

[14] K. Gorgolewski, C.D. Burns, C. Madison, D. Clark, Y.O. Halchenko, M.L.Waskom & S.S. Ghosh. Nipype: A flexible, lightweight and extensible neuroimaging data processing framework in python. Frontiers in Neuroinformatics 5(13)(2011). doi: 10.3389/fninf.2011.00013.

[15] E. Afgan, D. Baker, B. Batut, M. van den Beek, D. Bouvier, M. Čech, … & D. Blankenberg. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update, Nucleic Acids Research, 46(W1)(2018), W537–W54. doi: 10.1093/nar/gky379.

[16] K. Wolstencroft, R. Haines, D. Fellows, A. Williams, D. Withers, S. Owen, … C. Goble. The Taverna workflow suite: Designing and executing workflows of Web Services on the desktop, web or in the cloud. Nucleic Acids Research 41(W1)(2013), W557–W561. doi: 10.1093/nar/gkt328.

[17] E. Deelman, K. Vahi, J. Gideon, M. Rynge, S. Callaghan, P. Maechling,… & K.W. Pegasus. A Workflow Management System for Science Automation, Future Gener. Comput. Syst 46(C) (2015) 17–35. doi: 10.1016/j.future.2014.10.008.

[18] J. Köster & S. Rahmann. Snakemake—a scalable bioinformatics workflow engine. Bioinformatics 28(19)(2012), 2520–2522. doi: 10.1093/bioinformatics/bts480.

[19] P. Di Tommaso, M. Chatzou, E.W. Floden, P.P Barja, E Palumbo & C. Notredame. Nextflow enables reproducible computational workflows. Nature Biotechnology 35(4)(2017), 1546–1696. doi: 10.1038/nbt.3820.

[20] R. Filguiera, A. Krause, M. Atkinson & I. Klampanos. A Moreno dispel4py: A Python framework for data-intensive scientific computing. The International Journal of High Performance Computing Applications 31(4)(2017), 316–334. doi: 10.1177/1094342016649766.

[21] P. Moreno, L. Pireddu, P. Roger, N. Goonasekera, E. Afgan, M. van den Beek, … & S. Neumann. Galaxy-Kubernetes integration: Scaling bioinformatics workflows in the cloud. doi: 10.1101/488643.

[22] T. McPhillips, T. Song, T. Kolisnik, S. Aulenbach, K. Belhajjame, K. Bocinsky, … & B. Ludäscher. YesWorkflow: A user-oriented, language-independent tool for recovering workflow information from scripts. International Journal of Digital Curation 10(1)(2015). doi: 10.2218/ijdc.v10i1.370.

[23] M.D. Wilkinson, M. Dumontier, I.J. Aalbersberg, G. Appleton, M. Axton, A. Baak, … & B. Mons. The FAIR guiding principles for scientific data management and stewardship. Scientific Data 3(2016), Article No. 160018. doi: 10.1038/sdata.2016.18.

[24] J. Ison, M. Kalaš, I. Jonassen, D. Bolser, M. Uludag, H. McWilliam, J. Malone, R. Lopez, S. Pettifer & P. Rice. EDAM: An ontology of bioinformatics operations, types of data and identifiers, topics and formats. Bioinformatics 29(10)(2013), 1325–1332. doi: 10.1093/bioinformatics/btt113.

[25] S-A. Sansone, P. McQuilton, P. Rocca-Serra, A. Gonzalez-Beltran, M. Izzo, A.L. Lister & M. Thurston. The FAIRsharing Community FAIRsharing as a community approach to standards, repositories and policies. Nature Biotechnology 37(2019), 358–367. doi: 10.1038/s41587-019-0080-8.

[26] D. Garijo, P. Alper, K. Belhajjame, O. Corcho, Y. Gil & C. Goble. Common motifs in scientific workflows: An empirical analysis. Future Generation Computer Systems 36(2014), doi: 10.1016/j.future.2013.09.018.

[27] M.D. Wilkinson, S-A. Sansone, E. Schultes, P. Doorn, L.O. Bonino da Silva Santos & M. Dumontier. A design framework and exemplar metrics for FAIRness. Scientific Data 5 (2018)180118. doi: 10.1038/sdata.2018.118.

[28] P. Groth, H. Cousijn, T. Clark & C. Goble. FAIR data reuse – the path through data citation. Data Intelligence 2(2020), 78–86. doi: 10.1162/dint_a_00030.

[29] K. Chard, M. D'Arcy, B. Heavner, I. Foster, C. Kesselman, R. Madduri, A. Rodriguez, S. Soiland-Reyes, C. Goble, K. Clark, E.W. Deutsch, I. Dinov, N Price & A. Toga. I'll take that to go: Big data bags and minimal identifiers for exchange of large, complex datasets. In: IEEE International Conference on Big Data (Big Data), 2016, pp. 319–328. doi: 10.1109/BigData.2016.7840618.

[30] M. Herschel, R. Diestelkämper & H.B. Lahmar. A survey on provenance: What for? What form? What from? The VLDB Journal 26(6)(2017), 881–906. doi: 10.1007/s00778-017-0486-1.

[31] F.Z. Khan, S. Soiland-Reyes, R.O. Sinnott, A. Lonie, C. Goble & M.R. Crusoe. Sharing interoperable workflow provenance: A review of best practices and their practical application in CWLProv. GigaScience (in press) (2019).

[32] P. Alper, K. Belhajjame, V. Curcin & C.A. Goble. LabelFlow framework for annotating workflow provenance. Informatics 5(1)(2018), 11. doi: 10.3390/informatics5010011.

[33] D. Garijo, Y. Gil & O. Corcho. Abstract, link, publish, exploit: An end to end framework, for workflow sharing. Future Generation Computer Systems 75(2017), 271–283. doi: 10.1016/j.future.2017.01.008.

[34] J. Starlinger, S. Cohen-Boulakia, U. Leser. (Re)Use in Public Scientific Workflow Repositories. In: A. Ailamaki & S. Bowers (eds) Scientific and statistical database management. SSDBM 2012. Lecture Notes in Computer Science, 7338(2012). doi: 10.1007/978-3-642-31235-9_24.

[35] J. Starlinger, B. Brancotte, S. Cohen-Boulakia & U. Leser. Similarity search for scientific workflows. Proceedings of the VLDB Endowment 7(12)(2014), 1143–1154. doi: 10.14778/2732977.2732988.

[36] A.M. Smith, D.S. Katz & K.E. Niemeyer. FORCE11 Software Citation Working Group. Software citation principles. PeerJ Computer Science 2: e86, 2016. doi: 10.7717/peerj-cs.86.

[37] D. De Roure, C. Goble & R. Stevens. The design and realisation of the myExperiment Virtual Research Environment for social sharing of workflows. Future Generation Computer Systems 25(5)(2009), 561–567. doi: 10.1016/j.future.2008.06.010.

[38] K. Belhajjame, J. Zhao, D. Garijo, M. Gamble, K.M. Hettne, R. Palma, E. Mina, Ó. Corcho, J.M. Gómez-Pérez, S. Bechhofer, G. Klyne & C.A. Goble. Using a suite of ontologies for preserving workflow-centric research objects. Journal Web Semantics 32(2015), 16–42. doi: 10.1016/j.websem.2015.01.003.

[39] K. Plankensteiner, J. Montagnat & R. Prodan. IWIR: A language enabling portability across grid workflow systems. In: Workshop on Workflows in Support of Large-Scale Science, 2011, pp. 97–106. doi: 10.1145/2110497.2110509.

[40] G. Terstyanszky, T. Kukla, T. Kiss, P. Kacsuk, A. Balasko & Z. Farkas. Enabling scientific workflow sharing through coarse-grained interoperability. Future Generation Computing Systems 37(2014), 46–59. doi: 10.1016/j.future.2014.02.016.

[41] M. Haendel, A. Su, J. McMurry, C.G. Chute,C. Mungall,B. Good, ... & T. Conlin. FAIR-TLC: Metrics to assess value of biomedical digital repositories: Response to RFI NOT-OD-16-133. doi: 10.5281/zenodo.203295.

[42] D.S. Katz. Transitive credit as a means to address social and technological concerns stemming from citation and attribution of digital products. Journal of Open Research Software 2(1)(2014), e20. doi: 10.5334/jors.be.

[43] F. Casati, S. Ceri, B. Pernici & G. Pozzi. Workflow evolution. Data and Knowledge Engineering 24(3)(1998), 211–238. doi: 10.1016/S0169-023X(97)00033-5.

[44] C. Wroe, C. Goble, A. Goderis, P. Lord, S. Miles, J. Papay, P. Alper & L. Moreau. Recycling workflows and services through discovery and reuse. Concurrency and Computation Practice and Experience 19(2)(2006), doi: 10.1002/cpe.1050.

[45] H. Artaza, N.C. Hong, M. Corpas, A. Corpuz, R. Hooft, R.C. Jiménez, ... & D. Vaughan. Top 10 metrics for life science software good practices [version 1; peer review: 2 approved]. F1000Research 2016, 5(ELIXIR): 2000. doi: 10.12688/f1000research.9206.1.

[46] M. Taschuk & G. Wilson. Ten simple rules for making research software more robust. PLOS Computational Biology (2017). doi: 10.1371/journal.pcbi.1005412.
[47] F. da Veiga Leprevost, V.C. Barbosa, E.L. Francisco, Y. Perez-Riverol & P.C. Carvalho. On best practices in the development of bioinformatics. Software Front Genet 5(2014), 199. doi: 10.3389/fgene.2014.00199.
[48] G. Alterovitz, D. Dean, C. Goble, M.R Crusoe, S. Soiland-Reyes, A. Bell, et al. Enabling precision medicine via standard communication of HTS provenance, analysis, and results. Plos Biology 16(12)(2018), e3000099. doi: 10.1371/journal.pbio.3000099.
[49] V. Stodden, M. McNutt, D.H. Bailey, E. Deelman, Y. Gil, B. Hanson, M.A. Heroux, J.P.A. Ioannidis & M. Taufer. Enhancing reproducibility for computational methods Science 354(6317)(2016), 1240–1241. doi: 10.1126/science.aah6168.